Authors' responses to reviewer comments for "Meta-learning how to forecast time series"

Manuscript number FOR-21-0572

We would like to thank the Editor, Associate Editor and the anonymous reviewers for reviewing our paper, and for their insightful comments and suggestions. In this revision, we have addressed all the comments raised by the reviewers, and we have provided a point-by-point response to each comment.

Responses to Referee(s)' Comments

The subject of the work is important, the work is mature, well written, and the experimental part is extensive. To improve further the paper quality, the Authors can take into account the comments below.

(1) Page 7/line 5: what do you mean by the test set? Usually the test set (separate from the training and validation sets) cannot be used for training, optimization and selection of the model. Clarify, please.

Unfortunately, we were unable to find the place that corresponds to this comment. However, we have clarified this point in the revised manuscript in Section 2.3 and highlighted it below. See also our response to comment 3 below. Note that with the models we have used, there is no need for a validation set.

The training set is used to estimate the parameters of a forecasting model. Based on this fitted model, we generate forecasts over the test set and compare them against the test set values. Since the test data are not used for model fitting, this practice provides a reliable indicator to evaluate how well the model makes accurate forecasts on new data.

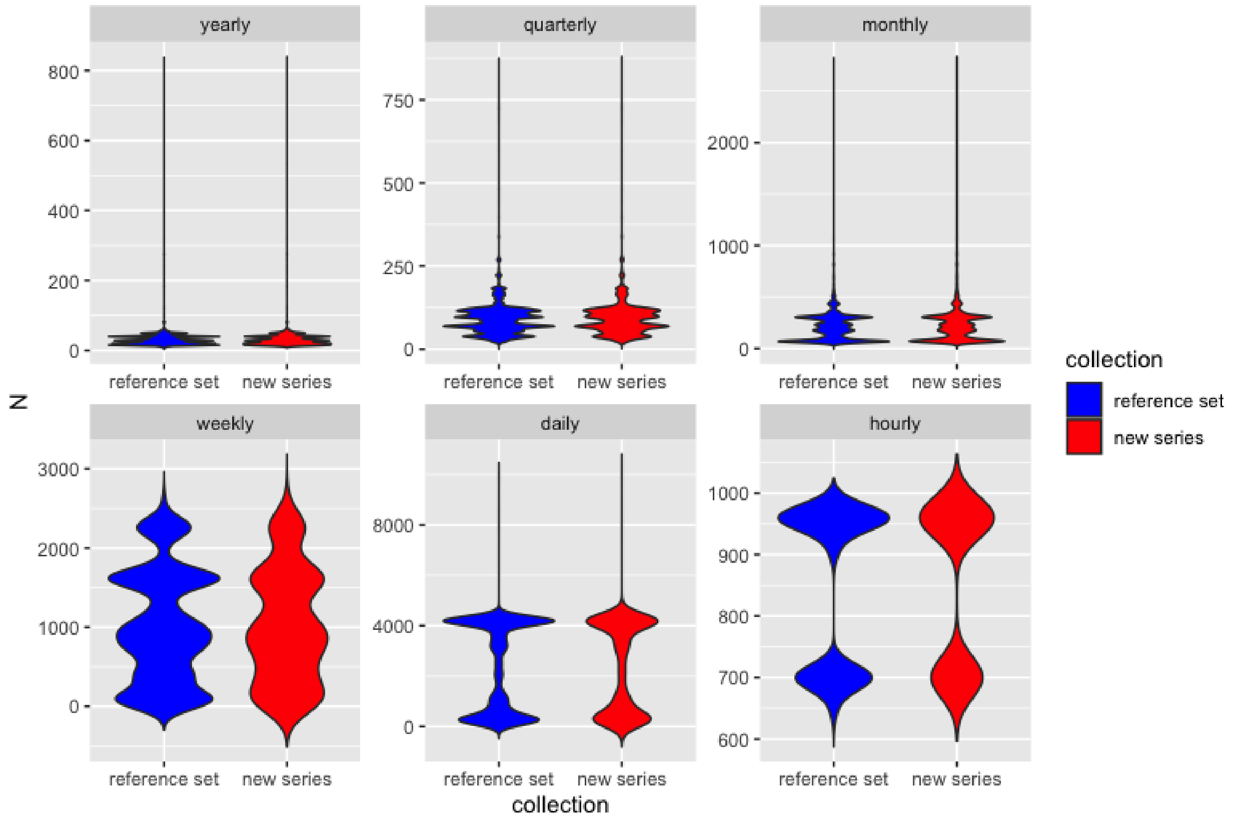
(2) Fig 3. It looks like red "feature calculation" block does the same work as blue "input-features" block. Thus unify the names of these blocks so as not to cause confusion.

We have changed the wording in Fig 3 - blue part "input-features" to "feature calculation" to unify the names and avoid confusion.

(3) 9/54: Explain how the time series are split into training and test part. How sensitive is the proposed approach to this split?

We include length as one of our features, but the remaining features are asymptotically independent of the length of the time series (i.e., they are ergodic), and they are independent of scale. We have added the following to section 3.1

In our research, since we used the M-competition data, we used the training and test sets specified by the M-competition organizers. In the M-competitions the training set length for series varies, however, the test length for the time series is fixed for each frequency level. The numbers of observations in the test periods are 6 for yearly, 8 for quarterly, 18 for monthly, 13 for weekly, 14 for daily, and 48 for the hourly series. Use of this standard test set specified by the M-competitions allows to compare our results with the results of other studies. Figure 4 shows the distributions of length of the training period of the M-competition data and the simulated data used in the study.



Furthermore, we added following to the conclusions section

Our method is sensitive to the lengths of the time series because we employed length as an input feature. We found that the algorithm chooses simple models (such as random walks or white noise) for shorter time series, while heavily parameterized models are more likely to be chosen for long time series. PDP curves of length revealed that there is a threshold beyond which the probability of choosing various models in response to training set length starts to plateau.

(4) 9/55: Fitting the model is related to its training and optimization (selection of hyperparameters). Moreover, the model performance depends on the feature selection/engineering. I suppose that these all tasks are performed by the "fit models" block in Fig. 3. You omit the topic of model optimization including feature selection/engineering, which can be very complex and time consuming process due to huge search space for same models. Describe this topic to make the reader aware of its importance.

Our feature set was based on prior studies, and we did not introduce any new features, or undertake any feature selection, as part of our algorithm. We have now clarified this in Section 3.3, as follows, with additions in blue:

Although the FFORMS framework could use any time series features, our current implementation uses the features shown in **??**; these are defined in the Appendix. These features were chosen based on their demonstrated utility in @kang2017visualising and @MonManEtAl2020.

\textcolor{teal}{We have also added some further information about this topic in the discussion and conclusion sections.

Hyperparameter tuning, feature selection, and feature engineering are essential tasks in machine learning that can have a significant effect on the performance of the machine learning algorithm. In this study, we have selected features that are known to have been helpful in other studies, and we have used default settings for the various hyperparameters. It is expected that further improvements in the algorithm could be achieved by carefully engineering new features, by removing some of the features we have included, and by tuning the hyperparameters of the random forest classifier.

(5) Algorithm 1: One of the key hyperparameters of random forests, in addition to the number of trees and the number of features to be selected at each node, is the minimum number of leaf node observations or its equivalent. You don't mention this hyperparameter at all. Why?

We now mention these parameters in Section 3.5. The additions we made are highlighted in blue. The relevant part of the paper now reads as follows:

The number of trees ntree is set to 1000, and the number of randomly selected features `k` is set to be one third of the total number of features available, while the minimum number of leaf node observations is 1. The number of trees is set to 1000 to reduce the time in both the online and offline phases of the algorithm.

(6) 11/5: "One approach is to fit models to the time series in the reference set, and then use those models as data generating processes to obtain similar time series." Clarify this issue.

Please see section 3.2. We have now revised this paragraph in an attempt to make it clearer. The relevant part of the paper now reads as follows (the new additions are highlighted in blue):

There are several options that could be used to produce simulated series that are similar to those in the population. One approach is to fit models to the time series in the observed sample, and then use those models as data generating processes to obtain similar time series. This could be done, for example, using exponential smoothing models, ARIMA models or other modelling functions in the forecast package in R [@forecast]. For each fitted model, we can simulate multiple time series with similar characteristics to the observed data. An alternative approach is to generate new time series with similar *features* from those found in the reference set [@kang2020gratis], using the gratis package in R [@Rgratis]. Either way, these simulated series are produced in the offline phase of the algorithm, so the computational time in producing them is of no real consequence.

(7) Time series augmentation looks like important component of the proposed framework and should be described in detail.

We have addressed this above following comment 6. We futher added the following to Section 3.2

Data augmentation is used to expand the amount of training data by adding significantly changed versions of either existing data, or brand-new synthetic data that is derived from existing data. This helps to increase the generalizability of the algorithm.

(8) 13/46: "incorporating class priors into the RF classifier". Could you explain this, please.

This is mentioned in our paper under Section 3.5. We added a new sentence to clarify this point. The relevant part of the paper now reads as follows:

The likelihood that a given class will appear in the bootstrap samples is represented by the class priors, which are introduced through the option `classwt`. We use the reciprocal of class size as the class priors.

(9) Many forecasting models, especially machine learning ones such as NNs, have a stochastic nature, which translates into different results for the same input data in different training sessions. This may have a negative impact on the choice of the best model. How does your proposed approach deal with such problem? Explain, please.

We have now added the following comment to the concluding section.

Several machine learning-based forecasting models such as NNs, have a stochastic estimation procedure, which leads to different results for the same input data in different training sessions, and this can affect the choise of best model. In such cases, one could fit many such models with different random seeds and use the average of the forecast error measure to select the best forecasting model when labelling time series.